Motion JPEG for video work

An in-depth guide by Space Banana

There's a huge variety of video encoding formats and encoders that were tailored for video work, such as video editing, color grading, VFX and more. One format that isn't very common, however, is Motion JPEG, but is it because it's bad for working with video? Not at all.

In this article we will see the strong and weak points of MJPEG for video work and when you should use this encoding format. The guide focuses on encoding video with FFmpeg and all parameters and arguments mentioned are for FFmpeg. The encoder library used is "mjpeg".

What matters for video work

For manipulating and editing video footage, we need to keep in mind some technical encoding properties:

Data preservation and quality

We want our video format to support high data preservation through high or custom bitrates. This generally isn't a problem as most video encoding formats support either one or the other. We also want our format to support pixel formats with low subsampling, such as YUV 4:2:2 and 4:4:4 instead of YUV 4:2:0, or even RGB, which does not have any subsampling, just like 4:4:4. Color subsampling reduces our color resolution and we lose color information. This is mostly visible when we color grade our footage, edit it or zoom-in. It's also easier to notice this on low resolution videos. If you want to capture high bit depth video, you should also keep in mind if your encoding format and encoder support high bit depth encoding.
Motion JPEG does most of these just fine. We can specify a target quality factor, and the lowest we can go is very high quality, providing little data loss while also not using tremendous data rates, unlike other intraframe formats like Cineform. Motion JPEG also supports YUV 4:4:4 and other subsampling choices, but it does not support RGB or high bit depth. For most use cases, this is fine.

Intraframe and lightweight compression

Like many encoding formats that are appropriate for manipulation and editing, Motion JPEG uses a lightweight intraframe compression. This means that Motion JPEG does not share identical pixel data between frames, while also having lightweight compression. This is crucial for a lightweight and performant video decode, which will impact the ultimate performance during editing.

How MJPEG compares to other formats

There are ups and downs to using Motion JPEG rather than using other formats.

MJPEG is much faster to decode than an H.264 encode that is optimized for playback performance, although it will use higher bitrates for similar quality.

Video encoders and encoding formats that are specifically made for professional video work will have high bit depth support, unlike Motion JPEG. They will also be more lightweight to playback, however they will have an even lower compression efficiency and higher bitrate for similar quality levels. Motion JPEG is then a good middle-ground between lightweight playback and file size.

Motion JPEG is also very fast to encode, even faster than other professional video formats.

To confirm this, I performed an extensive benchmark on FFmpeg where I compare the encoding speed between the Motion JPEG encoder "mjpeg" and other encoders of other formats. All tests encode the same input video, with the same output length.

System setup:

- OS: NixOS Tapir (23.11)
- Kernel version: 6.6.8
- CPU: Intel i7–1165G7, clock speed fixed at 2.5GHz
- FFmpeg version: 6.0

All encode samples use the following common CLI arguments:
`-y -hide_banner -t 35 -c:a copy`

Besides those arguments, each encode test uses encoder-specific arguments:

- **MJPEG quality 20**
  - `-c:v mjpeg -q 20 -qmax 20 -pix_fmt yuv420p`
- **MJPEG quality 1**
  - `-c:v mjpeg -q 1 -qmin 1 -pix_fmt yuv420p`
- **MJPEG quality 30**
  - `-c:v mjpeg -q 30 -qmax 30 -pix_fmt yuv420p`
- **MJPEG quality 80**
  - `-c:v mjpeg -q 80 -qmax 80 -pix_fmt yuv420p`
- **x264 veryfast (crf 30)**
  - `-c:v libx264 -preset:v veryfast -crf 30 -g 0 -pix_fmt yuv420p`
- **x264 ultrafast (crf 30)**
  - `-c:v libx264 -preset:v ultrafast -crf 30 -g 0 -pix_fmt yuv420p`
- **Utvideo**
  - `-c:v utvideo -pix_fmt yuv420p`
- **Cineform (quality 12)**
  - `-c:v cfhd -quality 12 -pix_fmt yuv422p10le`
- **Cineform (quality 0)**
  - `-c:v cfhd -quality 0 -pix_fmt yuv422p10le`
- **DNxHR HQ**
  - `-c:v dnxhd -profile:v dnxhr_hq -pix_fmt yuv422p`

The encoding speed is measured in frames encoded per second. The results are as follows:

- MJPEG quality 20: 107 FPS
- MJPEG quality 1: 88 FPS
- MJPEG quality 30: 105 FPS
- MJPEG quality 80: 107 FPS
- x264 veryfast (crf 30): 78 FPS
- x264 ultrafast (crf 30): 102 FPS
- Utvideo: 85 FPS
- Cineform (quality 12): 68 FPS
- Cineform (quality 0): 58 FPS
- DNxHR HQ: 72 FPS

Among the chosen encoders, MJPEG and x264 under the ultrafast preset are your best choices for lightweight encoding. Between MJPEG and x264 ultrafast, note that MJPEG is also faster to decode and uses reasonable bitrates.

Surprisingly, DNxHR and Cineform performed worse than expected in encoding speed.

An important detail is that all videos are encoded in 8bit/channel YUV 4:2:0 except the Cineform tests, which are encoded in 10bit YUV 4:2:2. This raises the encode overhead a bit. The reason all other tests where chosen as 4:2:0 is to preserve the color sampling of the original video and Cineform does not support 8bit 4:2:0 encoding.

MJPEG seems to perform worse with the best quality targets (in this case, quality value of 1) compared to lower quality targets, but this could have been an occasional error in measurement. It still performs well regardless.

Utvideo performs well, surprisingly better than DNxHR and Cineform.

Which format to choose

There are many solutions for recording footage to be used in video work, but among the formats and encoders I used for testing, here's the general recommendations:

- For the absolute encoding performance, use MJPEG or x264 ultrafast. If you require fast playback performance, use MJPEG.
-
- For high bit depth encodes, use x264, DNxHR or Cineform.
-
- For RGB encoding, use x264 or Utvideo.
-
- For lossless encoding, use x264 or Utvideo. Choose Utvideo for faster playback, and x264 for smaller bitrates (under slower presets than ultrafast).
- For very high fidelity and playback speed, regardless of encoding speed and file size, use Cineform. DNxHR is also a good choice, but it

underperforms in quality at low resolutions, the encoder scales better at higher resolutions.

High-quality MJPEG

MJPEG, just like encoding a JPEG image, relies on a quality factor to determine how much data it uses or tosses away. The default value in FFmpeg isn't ideal, we should change that.

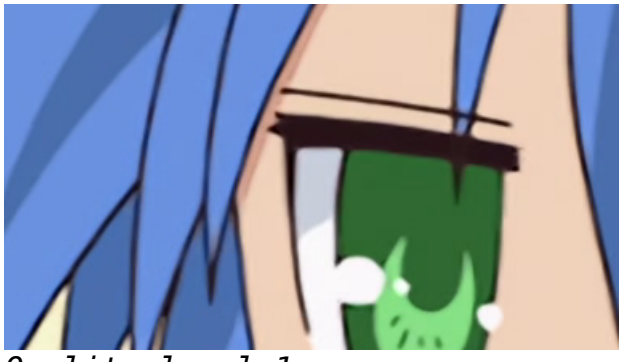
On FFmpeg, you can determine MJPEG's quality target with the "-q" parameter, as well as "-qmin" and "-qmax" for setting up the range of accepted values. The lower the quality value is, the higher the video quality and data preservation. For the highest quality, you can use "-qmin 1 -q 1". It's important that you specify "-qmin" here because the default minimum value is 2. At the quality level of 1, your video is "perceptually lossless", meaning what, while your encode is not lossless, it looks indistinguishable unless you zoom in and compare the pixels side by side. The bitrate is pretty reasonable at these quality levels, but if you want a smaller footprint, you can raise the quality value. Don't go too high or the video quality will be noticeably lower.

The quality outcome is constant throughout the whole video, thanks to the intraframe encoding of Motion JPEG. Identical pixel data isn't shared between frames, and so, while you won't have a quality raise in scenes with low motion, you also won't have a severe quality drop in scenes with significant or fast motion.

To test how much quality degrades over the levels, I performed a benchmark. The benchmark is very similar to the previous one in this article, with just a few small differences. All videos are encoded in MJPEG YUV 4:2:0, with only the quality level varying.

6 encode tests are made, with these respective quality levels: 1, 2, 3, 4, 5, and 10. A frame of each video is captured, with a crop to focus on a more zoomed-in area. The output videos are 35 seconds in length, and the audio data is preserved.

The source video is an episode of the anime Lucky Star, encoded in H.265 at 10bit/channel YUV 4:2:0. Here are the results:



Quality level 1

Quality level 2

Quality level 3

Quality level 4

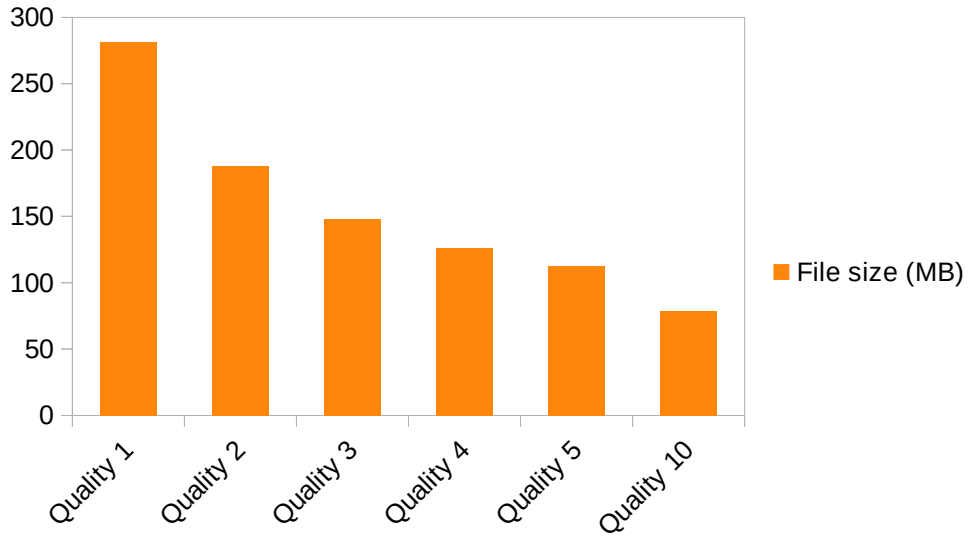
Quality level 5

Quality level 10

At first glance, they all look the same. If you look closely without zooming-in, you will notice that quality level 10 starts to noticeably look worse. However, when you zoom in images, you notice that they present a noticeable quality degradation. Sure, it's not visible without zooming in, but that perspective is more appropriate for when we want to deliver our final video. If we want to archive, manipulate, edit or color grade our videos, it's important they don't have this loss of quality. Try to edit a video with low quality footage that at first seems just fine, and your color grading, your transformations, scales, zoom-ins, distortions will easily reveal the low quality of the video, while also giving you less room for manipulating.

For this reason, it's not recommended you go above quality 3 for your footage.

As for the difference in file sizes, here's the results:



File size lower linearly, with the biggest difference being from quality 1 to quality 2. Quality loss is proportionate to reduction in file size, which is good and predictable. If you are tight on space but still want to encode high-fidelity videos, quality level 2 is ideal.

Note that the output videos contain an audio channel copied from the input video. Without this channel, the output videos would be smaller. This does not affect the benchmark's accuracy as we are comparing relative difference in file size throughout quality levels.